

本周写了一些代码，实现了 SDL 描述的基本形状，包括 up, Up, down, Down, stable, zero, appear, disappear。感觉奇怪的是 zero、appear 有时候会代表相同的状态，对于具体的数据也无法区分。仔细看论文后，作者在实现时对二者做了区分，通常 appear 只在时间序列开始的时候起作用。代码能够对时间序列做出判断，给定一段时间序列能够根据 SDL 指定的规则将数据转换成上述状态。目前正在做的工作是按照这些状态对数据建立索引。先前已经通过实验明确了如何在文件中组织并存储索引，只要按照 SDL 中的规则构建索引结构就能实现时间序列的快速查询。

另外，我还查看了关于 crowdsourcing 的两篇论文，作者都是 Stanford 的 Aditya Parameswaran。其中“Answering Queries using Humans, Algorithms and Databases”是作者对 crowdsourcing 中 human computation、algorithm computation 以及 data 三者结合的总体描述，重点介绍如何在 crowdsourcing 中有效地协调 uncertainty, cost and performance，其方法是数据库中的查询优化。作者首先构建了查询模型，并说明了在此模型上如何解决 uncertainty, cost 和性能问题，这篇论文发表于 CIDR 2011。值得一提的是，作者在文中列举了一个典型的任务实例：从给定的大量图片中寻找对应酒店的照片。要求图片不能是黑的，不能有版权问题，照片中要么显示了酒店的食品服务信息要么显示带有酒店名称的外景。这个任务可以分为多个子任务，这些任务中有的与算法有关、有的与 crowdsourcing 有关、有的则与任务提出者有关。判断图片是否是黑的可以由算法完成，而选择照片可以由 crowdsourcing 完成，对于最后返回的结果则需要任务提出者判断。

2012 年 PVLDB 上发表了另一篇论文“Deco: Declarative Crowdsourcing”。作者针对上述 vision 开发了一个原型系统 Deco。Deco 定义了一个更加具体的数据模型，在模型中作者描述了整个数据查询过程，如下图。

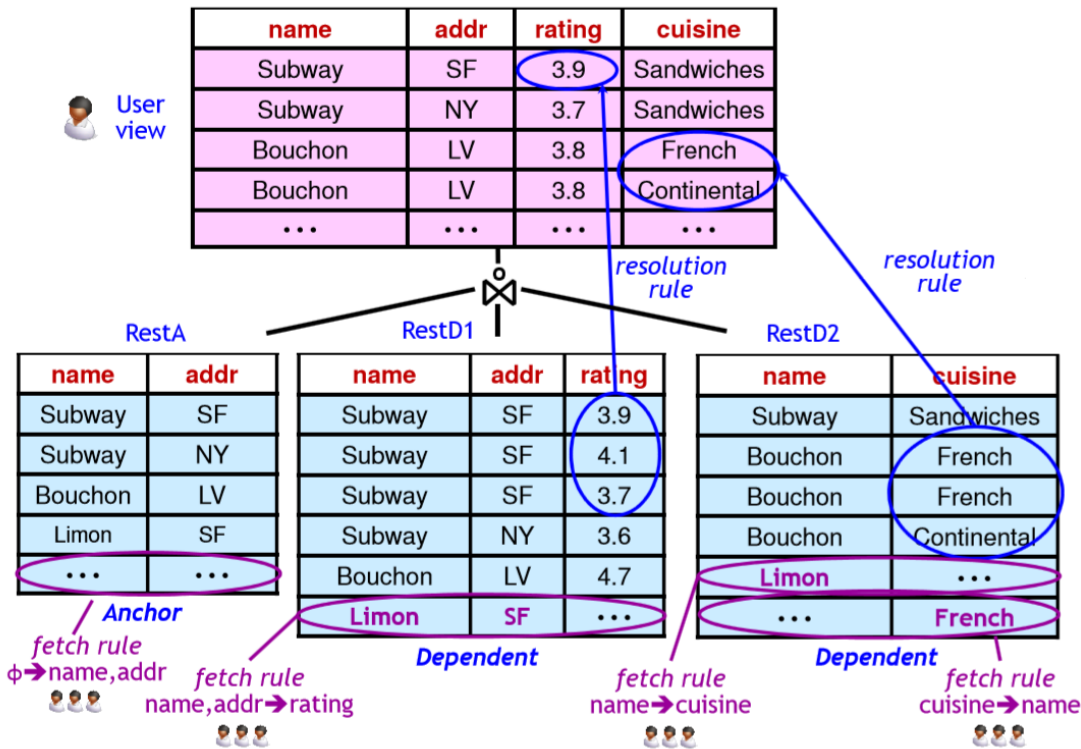


Figure 1: Components of the Deco Data Model

图中 RestA, RestD1, RestD2 为 raw tables，是原始数据，上面的表为用户查询之后的结果。Deco 就是要解决如何从下面的三个表生成最终的用户看到的视图。为此作者定义了两类表：

Anchor table 和 Dependent table 和两类规则：fetch rule 和 resolution rule。Anchor table 中的都是具体的实体，而 dependent table 中的列描述的是实体的属性。Fetch rule 是要求 crowdsourcing workers 去完成的，如让 workers 填写 restaurant 的名称和地址，或者指定名称和地址要求对 restaurant 填写 rating 值，或者根据 cuisine 填写 restaurant 的名称等。而 resolution rule 则是系统自动完成的，主要有两种：

- 1、根据相同实体不同的 rating 值计算最终用户视图中的 rating 值，计算平均值。
- 2、要求每个 restaurant 都具有 raw table 中所有的 cuisine，而且相同 restaurant 不能有重复的 cuisine。

在数据模型之后，作者介绍了整个模型的设计和实现过程，并对原型系统做了性能评测。

在 crowdsourcing 问题中，重点提到了 human computation，人在其中起到非常重要的作用。Luis von Ahn（Carnegie Mellon University）在“Game with a purpose”一文中提到未来可以设计一些游戏由 crowdsourcing 来完成很复杂的任务，比如标注图片、翻译、安全监控、为人在网上查询资料、总结文章等。在可视化中，我觉得可以考虑的有两点：一是延伸 Deco 模型，将系统扩展为可视化交互界面；二是分解任务（使得任务粒度合适而且价格不超过上限）和对任务结果进行判别是否达标。

下周实现 SDL 的索引，并做一定数据量的初步测试。